OC Pizza

IT Consulting & Development

DOSSIER DE SPECIFICATIONS TECHNIQUES

« Projet 6 : Concevez la solution technique d'un système de gestion de pizzeria »

Version 1.6 – 28/09/2020

Résumé

L'objectif du document est de proposer une description du domaine fonctionnel, de décrire les différents composants du système, les composants externes utilisés par celui-ci, leurs interactions et de faire la description de l'organisation physique de ces composants.

Loïc ROMERO

Analyste-programmeur

loic.romero1@gmail.com

OC Pizza

IT Consulting & Development

Table des matières

1 - VERSIONS

Auteur	Date	Description	Version
Loïc ROMERO	07/09/2020	Création du document.	1.0
Loïc ROMERO	21/09/2020	Ajout du plan.	1.1
Loïc ROMERO	23/09/2020	Ajout des schémas.	1.2
Loïc ROMERO	24/09/2020	Rédaction (Chap. : 2.1 – 2.2 – 2.3 – 3.1)	1.3
Loïc ROMERO	25/09/2020	Rédaction (Chap. : 3.3)	1.4
Loïc ROMERO	26/09/2020	Rédaction (Chap. : 4.1 – 4.3)	1.5
Loïc ROMERO	28/09/2020	Rédaction (Chap. : 5 – 6)	1.6

2 - INTRODUCTION

2.1 - Objet du document

Le présent document constitue le **dossier de conception technique** du « Projet 6 : Concevez la solution technique d'un système de gestion de pizzeria ».

Il fait suite au dossier de conception fonctionnel. Il est issu des conversations avec le client.

L'objectif du document est de **décrire les méthodes, procédés et technologies sélectionnés** pour faire face aux contraintes de réalisation du projet. Il s'agit d'obtenir l'accord du client **avant le début de la phase de réalisation**.

Toute modification en cours de réalisation, devra au préalable être approuvée par les parties concernées et tracées dans ce même document.

Les éléments présents dans le dossier, permettrons de comprendre la **nature, l'organisation et les relations entre les composants** de la solution informatique précédemment décrite dans le dossier de conception fonctionnel.

Un rappel du contexte, des enjeux et des objectifs issus du besoin client, OC Pizza, sera rédigé.

Puis, par le biais de la méthodologie **UML** seront présentés :

- *Le diagramme de classes.*
- *Le diagramme de représentation du modèle physique de données (MPD).*
- *Le diagramme de composants.*
- *Le diagramme de déploiement.*

Ces schémas seront détaillés avec précision.

Le présent *document, «* **Dossier de spécifications techniques.pdf »** *est disponible à l'adresse suivante :*

- *https://github.com/ROL-1/P6-Solution-Technique*

2.2 - Références

Pour de plus amples informations, se référer également au **« Dossier de conception fonctionnelle V1.8.pdf »** disponible à l'adresse suivante :

- https://github.com/ROL-1/P4-OC-PIZZA

2.3 - Besoin du client

2.3.1 - Contexte

Le groupe « **OC Pizza** » (client) a contacté notre entreprise, « **IT Consulting & Development** » (prestataire) pour signifier son souhait de faire développer un système de gestion informatique pour l'ensemble des restaurants du groupe. Il s'agit de restaurants spécialisés dans les pizzas livrées ou à emporter.

La motivation de cette démarche est liée à la croissance du groupe. Possédant déjà 5 points de vente, il prévoit d'en ouvrir au moins 3 de plus d'ici **6 mois**.

Le système informatique actuel ne correspond plus aux besoins du groupe car il ne permet pas une gestion **centralisée** de toutes les pizzerias.

Les responsables ont besoin de **suivre** facilement ce qui se passe dans les points de vente.

Les livreurs ne peuvent pas indiquer « en live » que la livraison est effectuée.

2.3.2 - Enjeux et Objectifs

Le système informatique devra répondre aux besoins suivants :

Centraliser :

- Comptes clients uniques pour tous les types d'achat (en ligne – par téléphone – sur place).
- Interface d'achat reliée en ligne ou à la livraison.

Offrir de la **visibilité** :

- Site internet dédié aux clients (passer commande).

Organiser, **homogénéiser** :

- Gestion des commandes (réception – préparation – livraison).
- Relier les indicateurs de stocks à l'interface d'achat (savoir si une pizza est encore réalisable).
- Un client peut modifier ou annuler sa commande (avant la « préparation »)
- Proposer un aide-mémoire aux pizzaïolos indiquant la recette de chaque pizza.
- Définir un cycle de vie des commandes.
- Gestion administrative (comptes utilisateurs, indicateurs).

Offrir un **suivi** :

- En temps réel du statut des commandes.
- En temps réel du stock d'ingrédients.

Enfin, il sera impératif de proposer une solution pouvant être concrétisée dans un délai raisonnable pour permettre une prise en main par les responsables **avant** l'ouverture des nouveaux points de ventes d'ici **6 mois**.

3 - DOMAINE FONCTIONNEL

3.1 - Synthèse

Le **diagramme de classes** est un schéma réalisé, ici, avec la méthodologie **UML**. Les diagrammes de cas d'utilisations (du dossier de spécifications techniques) montraient le système du point de vue des acteurs et décrivaient les fonctionnalités attendues. Ici, il s'agit d'en montrer la **structure interne** qui permettra de réaliser ces fonctionnalités.

Il sert donc à présenter les classes et leurs **relations**. Qui, chacune, pourront être réutilisées dans plusieurs cas d'utilisation. C'est un diagramme dit « statique » car il fait abstraction des aspects temporels ou dynamiques.

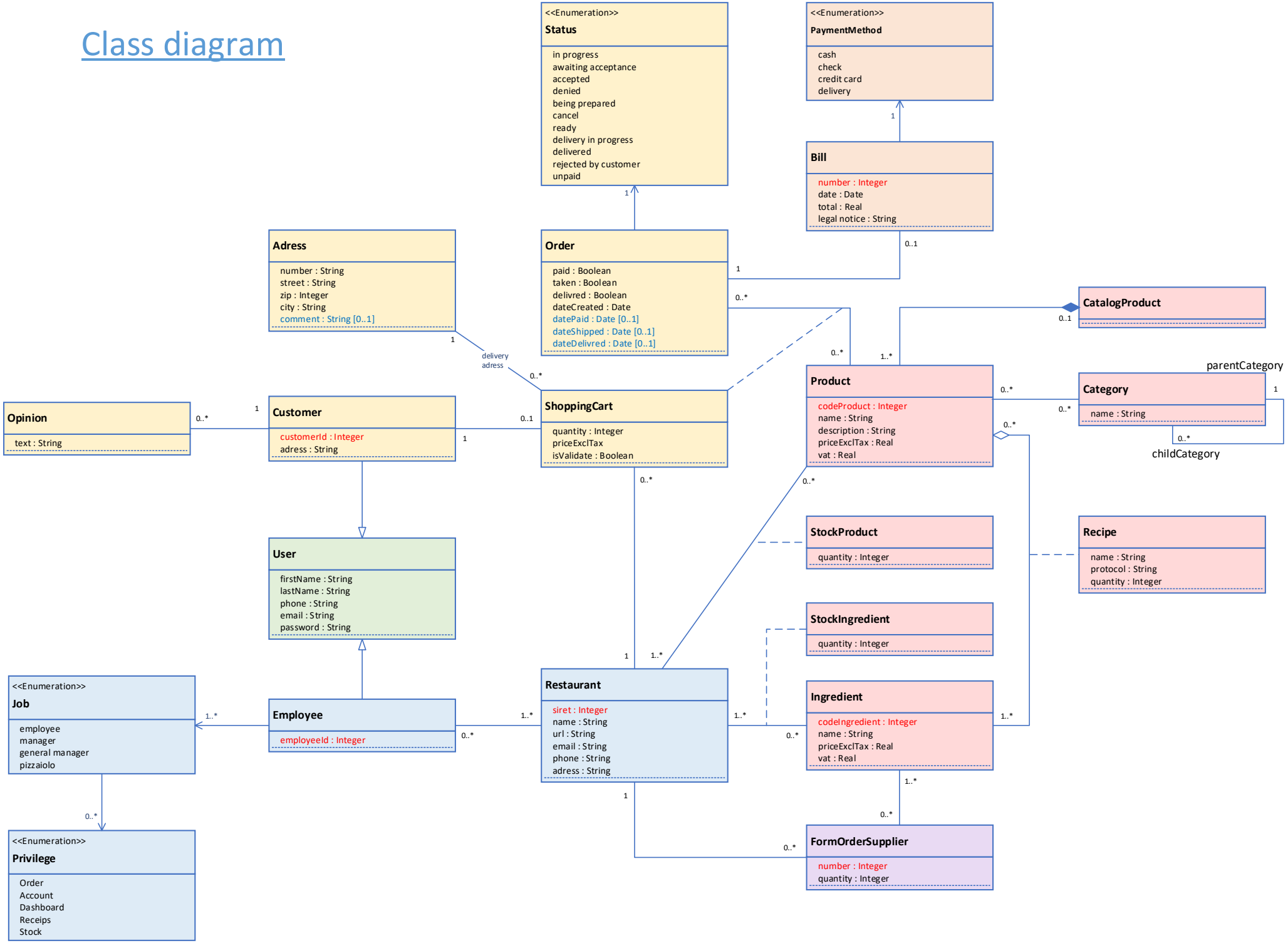
Chaque **classe** décrit les **responsabilités et le type** d'un ensemble d'objet. Elles permettront de modéliser le programme et ainsi de découper les tâches complexes en plusieurs travaux simples. Elles seront donc instanciées pour créer les objets de la **programmation orientée objet**. Il s'agit ici d'un **schéma fonctionnel**, aussi d'un point de vue technique, lors de rédaction du code, il sera possible de constater des découpages sensiblement différents.

Il est à noter que les méthodes n'apparaissent pas sur ce schéma. Les couleurs utilisées servent à faciliter la lecture en proposant des ensembles logiques, mais ne représentent pas une codification précise.

3.2 - Diagramme de classes

Schéma visible page suivante.

Class diagram

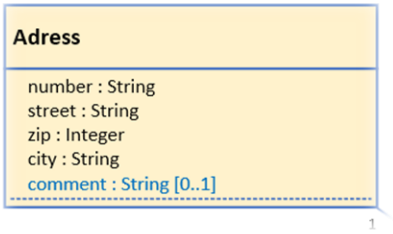


3.3 - Détail des classes

Chaque classe est détaillée, ci-dessous, dans deux tableaux :

- Attribut : décrit ce que contient chaque attribut de la classe. La colonne « PK » indique si l'attribut est une clef primaire.
- Relation : indique la multiplicité (ou cardinalité) de la relation entre la classe et celle indiquée dans la colonne « Classe », décrit la relation en précisant éventuellement sa nature (héritage, composition, agrégation).

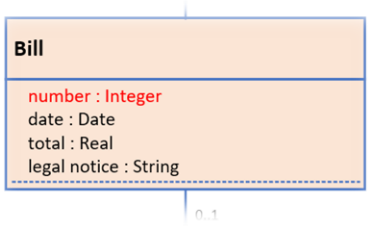
3.3.1 - Classe « Adress »



Attribut	Description	PK
number	*Numéro de porte.*	
street	*Nom de la rue.*	
zip	*Code postal.*	
city	*Nom de la ville.*	
comment	*Commentaire (pour les livraisons).*	

Relation	Classe	Description
0..1 - 1	*ShoppingCart*	*Une adresse est liée à aucun ou un panier.* *Un panier est lié à une unique adresse de livraison.*

3.3.2 - Classe « Bill »



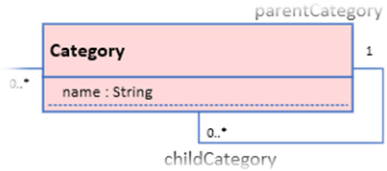
Attribut	Description	PK
number	*Numéro de facture unique.*	✓
date	*Défini si la commande a été acceptée ou non.*	
total	*Défini si la commande a été livrée ou non.*	
legal notice	*Date de création de la commande.*	

Relation	Classe	Description
0..1 - 1	*Bill*	*Une commande est liée à aucune ou une unique facture.* *Une facture est liée à une unique commande.*
0..* - 0..*	*Product*	*Une commande est liée à aucun ou plusieurs produits.* *Un produit est lié à aucune ou plusieurs commandes.*
-> 1	*Status*	*« Status » est de type « énumération » et liste donc des données pour un attribut de l'énumération « Order ». Permet de créer les indicateurs.*

3.3.3 - Classe « CatalogProduct »

Relation	Classe	Description
1..* - 0..1	*Product*	*« CatalogProduct » entretient une relation de composition avec « Product », l'ensemble des produits forment le contenu du catalogue produit.*

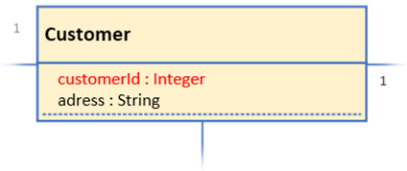
3.3.4 - Classe « Category »



Attribut	Description	PK
name	*Nom de la catégorie (ou sous-catégorie).*	

Relation	Classe	Description
0..* - 0..*	*Product*	*Une catégorie est liée avec aucun ou plusieurs produits.* *Un produit est lié avec aucune ou plusieurs catégories.*
0..1 - *	*Category*	*Une catégorie est liée a aucune ou plusieurs sous-catégories.* *Une sous-catégorie est liée à une unique catégorie.*

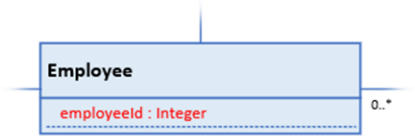
3.3.5 - Classe « Customer »



Attribut	Description	PK
customerId	*Numéro unique d'utilisateur « client ».*	✓
adress	*Adresse postale du client*	

Relation	Classe	Description
0..* - 1	*Opinion*	*Un compte client peut avoir aucun ou plusieurs avis.* *Un avis est toujours lié à un unique compte client.*
0..1 - 1	*ShoppingCart*	*Un compte client peut avoir aucun ou un panier actif.* *Un panier est toujours lié à un unique compte client.*
Héritage	*User*	*La table « Customer » hérite des mêmes attributs que ceux de la table « User ».*

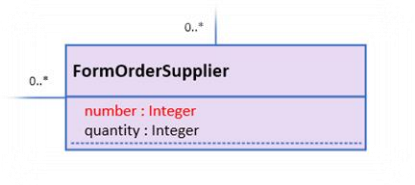
3.3.6 - Classe « Employee »



Attribut	Description	PK
employeeId	*Numéro unique d'utilisateur « employé » (ce terme désigne aussi bien un employé qu'un manager ou un membre de la direction).*	✓

Relation	Classe	Description
-> 1..*	*Job*	*Un compte employé possède un attribut dont les données sont listées dans la table « Job » de type « énumération ». Un compte-employé est lié à au moins un rôle (et éventuellement plusieurs).*
1..* - 0..*	*Restaurant*	*Un employé est lié à plusieurs ou au moins un restaurant.* *Le restaurant est lié à plusieurs ou aucun employé.*
Héritage	*User*	*La table « Employee » hérite des mêmes attributs que ceux de la table « User ».*

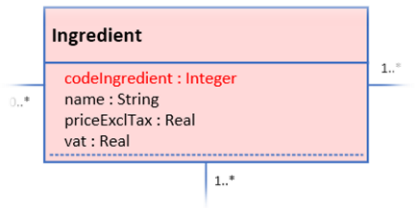
3.3.7 - Classe « FormOrderSupplier »



Attribut	Description	PK
number	*Numéro de bon de commande « fournisseur ».*	✓
quantity	*Défini la quantité d'ingrédient souhaitée pour le bon de commande.*	

Relation	Classe	Description
1..* - 0..*	*Ingredient*	*Un bon de commande est lié à au moins un ou plusieurs ingrédients.* *Un ingrédient est lié à aucun ou plusieurs bon de commande fournisseur.*
1 - 0..*	*Restaurant*	*Un bon de commande fournisseur est lié à uniquement un Restaurant.* *Un restaurant est lié à aucun ou plusieurs bons de commande.*

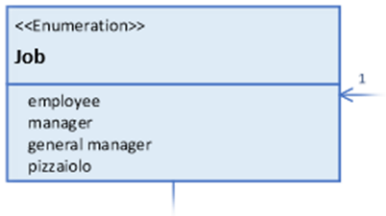
3.3.8 - Classe « Ingredient »



Attribut	Description	PK
codeIngredient	*Numéro d'ingrédient unique (code-barres).*	✓
name	*Nom de l'ingrédient.*	
priceExclTax	*Prix Hors Taxes du produit.*	
vat	*Montant de la T.V.A. pour l'ingrédient.*	

Relation	Classe	Description
1..* – 0..*	*FormOrderSupplier*	*Un ingrédient est lié à aucun ou plusieurs bon de commande fournisseur.* *Un bon de commande est lié à au moins un ou plusieurs ingrédients.*
0..* - 1..*	*Product*	*« Ingrédient » entretient une relation d'agrégation avec « Product », les produits sont créés avec les ingrédients.* *Un ingrédient est lié avec aucun ou plusieurs produits.* *Un produit est lié avec plusieurs ou au moins un ingrédient.*
0..* - 1	*Restaurant*	*Un ingrédient est lié à un et uniquement un restaurant.* *Le restaurant est lié à plusieurs ou aucun ingrédient.*

3.3.9 - Enumeration « Job »



	Donnée
employee	*Employé d'un restaurant (autre que pizzaïolo).*
manager	*Gérant d'un restaurant.*
general manager	*Membre de la direction du groupe.*
pizzaiolo	*Préparateur d'un restaurant.*

Relation	Classe	Description
1..* <-	*Employee*	*« Job » est de type « énumération » et liste donc des données pour un attribut de la classe Employee.*
-> 0..*	*Privilege*	*Un « Job » possède un attribut dont les données sont listées dans la table « Privilege » de type « énumération ». Un rôle est lié à plusieurs, un, ou aucun privilège.*

3.3.10 - Classe « Opinion »

Attribut	Description	PK
text	*Contient un champ de texte pour la rédaction d'un avis-client.*	

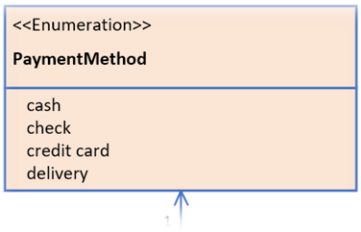
Relation	Classe	Description
1 – 0..*	*Opinion*	*Un avis est toujours lié à un unique compte client.* *Un compte client peut avoir aucun ou plusieurs avis.*

3.3.11 - Classe « Order »

Attribut	Description	PK
paid	*Défini si la commande a été payée ou non.*	
taken	*Défini si la commande a été acceptée ou non.*	
delivred	*Défini si la commande a été livrée ou non.*	
dateCreated	*Date de création de la commande.*	
datePaid	*Date du paiement de la commande.*	
dateShipped	*Date de la prise en charge par la livraison.*	
dateDelivred	*Date de livraison de la commande.*	

Relation	Classe	Description
0..1 - 1	*Bill*	*Une commande est liée à aucune ou une unique facture.* *Une facture est liée à une unique commande.*
0..* - 0..*	*Product*	*Une commande est liée à aucun ou plusieurs produits.* *Un produit est lié à aucune ou plusieurs commandes.*
-> 1	*Status*	*« Status » est de type « énumération » et liste donc des données pour un attribut de l'énumération « Order ». Permet de créer les indicateurs.*

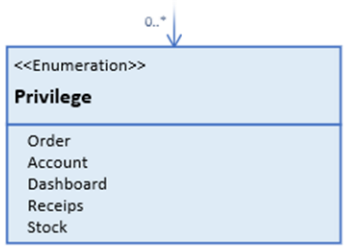
3.3.12 - Classe « PaymentMethod »



Attribut	Description	PK
cash	*Défini un moyen de paiement en liquide.*	
check	*Défini un moyen de paiement en chèque.*	
credit card	*Défini un moyen de paiement en carte de crédit.*	
delivery	*Défini que le paiement est effectué à la livraison.*	

Relation	Classe	Description
1 <-	*Bill*	*« PaymentMethod » est de type « énumération » et liste donc des données pour un attribut de la classe « Bill ». Défini le moyen de paiement de la commande.*

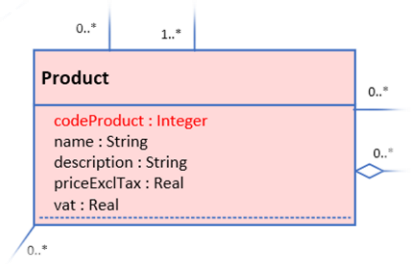
3.3.13 - Enumeration « Privilege »



	Donnée
order	*Privilèges liés à l'achat d'une commande.*
account	*Privilèges liés à la gestion des comptes.*
dashboard	*Privilèges liés au suivi des indicateurs.*
receips	*Privilèges liés à la gestion des recettes.*
stock	*Privilèges liés à la gestion des stocks*

Relation	Classe	Description
0..* <-	*Job*	*« Privilege » est de type « énumération » et liste donc des données pour un attribut de l'énumération « Job ».*

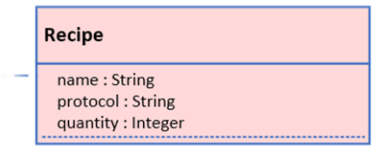
3.3.14 - Classe « Product »



Attribut	Description	PK
codeProduct	*Numéro de produit unique (code-barres).*	✓
name	*Nom du produit.*	
description	*Description du produit (commentaires).*	
priceExclTax	*Prix Hors Taxes du produit.*	
vat	*Montant de la T.V.A. pour le produit.*	

Relation	Classe	Description
0..1 – 1..*	*CatalogProduct*	*« CatalogProduct » entretient une relation de composition avec « Product », l'ensemble des produits forment le contenu du catalogue produit.*
0..* - 0..*	*Category*	*Un produit est lié avec plusieurs ou aucune catégorie.* *Une catégorie est liée avec aucun ou plusieurs produits.*
1..* - 0..*	*Ingredient*	*« Ingrédient » entretient une relation d'agrégation avec « Product », les produits sont créés avec les ingrédients.* *Un produit est lié avec plusieurs ou au moins un ingrédient.* *Un ingrédient est lié avec aucun ou plusieurs produits.*
0..* - 0..*	*Order*	*Un produit est lié à aucune ou plusieurs commandes.* *Une commande est liée à aucun ou plusieurs produits.*
1 - 0..*	*Restaurant*	*Un produit est lié à au moins un ou plusieurs restaurants.* *Un restaurant est lié à aucun ou plusieurs produits.*

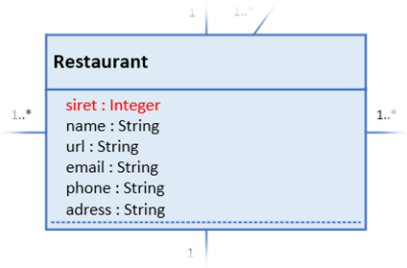
3.3.15 - Classe « Recipe »



Attribut	Description	PK
name	*Nom de la recette.*	
protocol	*Protocole de réalisation de la recette.*	
quantity	*Défini la quantité d'ingrédient souhaitée pour la recette.*	

Relation	Classe	Description
Association	*Product*	*La classe « Recipe » apporte ses attributs à l'association entre les deux classes « Product » et « Ingredient », définissant la recette.*

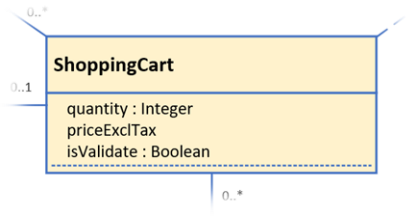
3.3.16 - Classe « Restaurant »



Attribut	Description	PK
siret	*Numéro unique de SIRET du restaurant.*	✓
name	*Nom du restaurant.*	
url	*Adresse du site du restaurant.*	
email	*Adresse email du restaurant.*	
phone	*Numéro de téléphone du restaurant.*	
adress	*Adresse postale du restaurant.*	

Relation	Classe	Description
0..* - 1..*	*Employee*	*Le restaurant est lié à plusieurs ou aucun employé.* *Un employé est lié à plusieurs ou au moins un restaurant.*
0..* - 1	*FormOrderSupplier*	*Le restaurant est lié à plusieurs ou aucun bon de commande.* *Un bon de commande est lié à un et uniquement un restaurant.*
0..* - 1..*	*Ingredient*	*Le restaurant est lié à plusieurs ou aucun ingrédient.* *Un ingrédient est lié à un ou plusieurs restaurants.*
0..* - 1	*Product*	*Le restaurant est lié à plusieurs ou aucun produit.* *Un produit est lié à au moins un ou plusieurs restaurants.*
0..* - 1	*ShoppingCart*	*Le restaurant est lié à plusieurs ou aucun panier.* *Un panier est lié à un et uniquement un restaurant.*

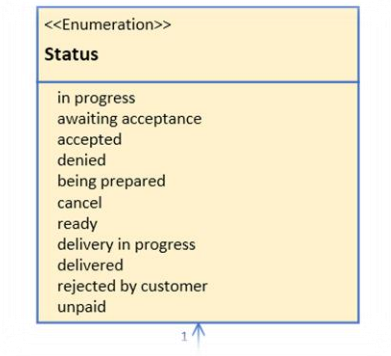
3.3.17 - Classe d'association « ShoppingCart »



Attribut	Description	PK
quantity	*Défini la quantité souhaitée pour chaque produit d'une commande.*	
priceExclTax	*Défini le prix Hors Taxe pour chaque produit d'une commande.*	
isValidate	*Défini si le panier a été validé par le client.*	

Relation	Classe	Description
1 - 1	*Adress*	*Un panier a une unique adresse de livraison.* *Une adresse de livraison est liée à aucun ou plusieurs paniers.*
1 - 0..1	*ShoppingCart*	*Un compte client peut avoir aucun ou un panier actif.* *Un panier est toujours lié à un unique compte client.*
Association		*Le panier apporte ses attributs à l'association entre les deux classes « Product » et « Order »*

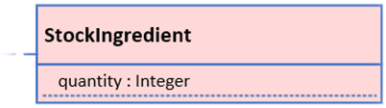
3.3.18 - Enumeration « Status »



	Donnée
in progress	*Commande créée.*
awaiting acceptance	*Commande en attente d'acceptation .*
accepted	*Commande acceptée par le restaurant.*
denied	*Commande rejetée par le restaurant.*
being prepared	*Commande en cours de préparation.*
cancel	*Commande annulée par le restaurant.*
ready	*Commande prête, en attente de prise en charge par le service de livraison.*
delivery in progress	*Commande en cours de livraison.*
delivered	*Commande livrée.*
rejected by customer	*Commande rejetée par le client.*
unpaid	*Commande non payée par le client.*

Relation	Classe	Description
1 <-	*Order*	*« Status » est de type « énumération » et liste donc des données pour un attribut de l'énumération « Order ». Permet de créer les indicateurs.*

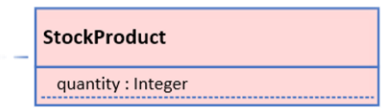
3.3.19 - Classe « StockIngredient »



Attribut	Description	PK
quantity	*Défini la quantité d'ingrédient en stock par restaurant.*	

Relation	Classe	Description
Association		*La classe « StockProduct » apporte ses attributs à l'association entre les deux classes « Ingredient » et « Restaurant », définissant le stock ingrédient.*

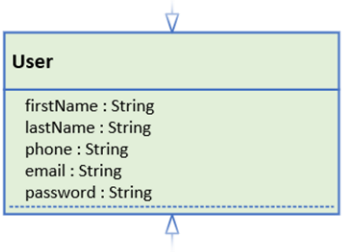
3.3.20 - Classe d'association « StockProduct »



Attribut	Description	PK
quantity	*Défini la quantité de produit en stock par restaurant.*	

Relation	Classe	Description
Association		*La classe « StockProduct » apporte ses attributs à l'association entre les deux classes « Product » et « Restaurant », définissant le stock produit.*

3.3.21 - Classe « User »



Attribut	Description	PK
firstName	*Prénom de l'utilisateur*	
lastName	*Nom de famille de l'utilisateur*	
phone	*Numéro de téléphone de l'utilisateur*	
email	*Email de l'utilisateur*	
password	*Mot de passe de l'utilisateur*	

Relation	Classe	Description
Héritage	*Customer*	*La table « Customer » hérite des mêmes attributs que ceux de la table « User ».*
Héritage	*Employee*	*La table « Employee » hérite des mêmes attributs que ceux de la table « User ».*

4 - MODELE PHYSIQUE DE DONNEES

4.1 - Synthèse

Inspiré du diagramme de classes, le modèle physique de données (**MPD**) présente les **tables** utilisées pour **stocker les données** du système informatique. Il s'agit de la représentation de la **base de données (BDD)**. Le MPD sert à implanter la base de données (grâce au reverse engineering) et lui est donc fidèle. Les tables sont constituées d'**attributs**, correspondant aux colonnes d'un tableau, et d'**instances**, correspondant aux lignes. Les relations entre les tables sont identifiées par la notation « patte d'oie ».

Des **différences** apparaissent par rapport au diagramme de classes lorsque l'expression fonctionnelle ne correspond pas aux **besoins techniques** de la création de cette architecture de stockage de données.

Notamment, le panier (« ShoppingCart ») n'apparait pas directement car sa gestion est déléguée à la partie site web (type cookies/API JavaScript). On notera également la fusion des classes « Customer » et « Employee » dans une unique table « User », la différentiation pouvant se faire par l'attribution de rôle spécifiques.

Chaque table sera détaillée dans la partie 4.3.

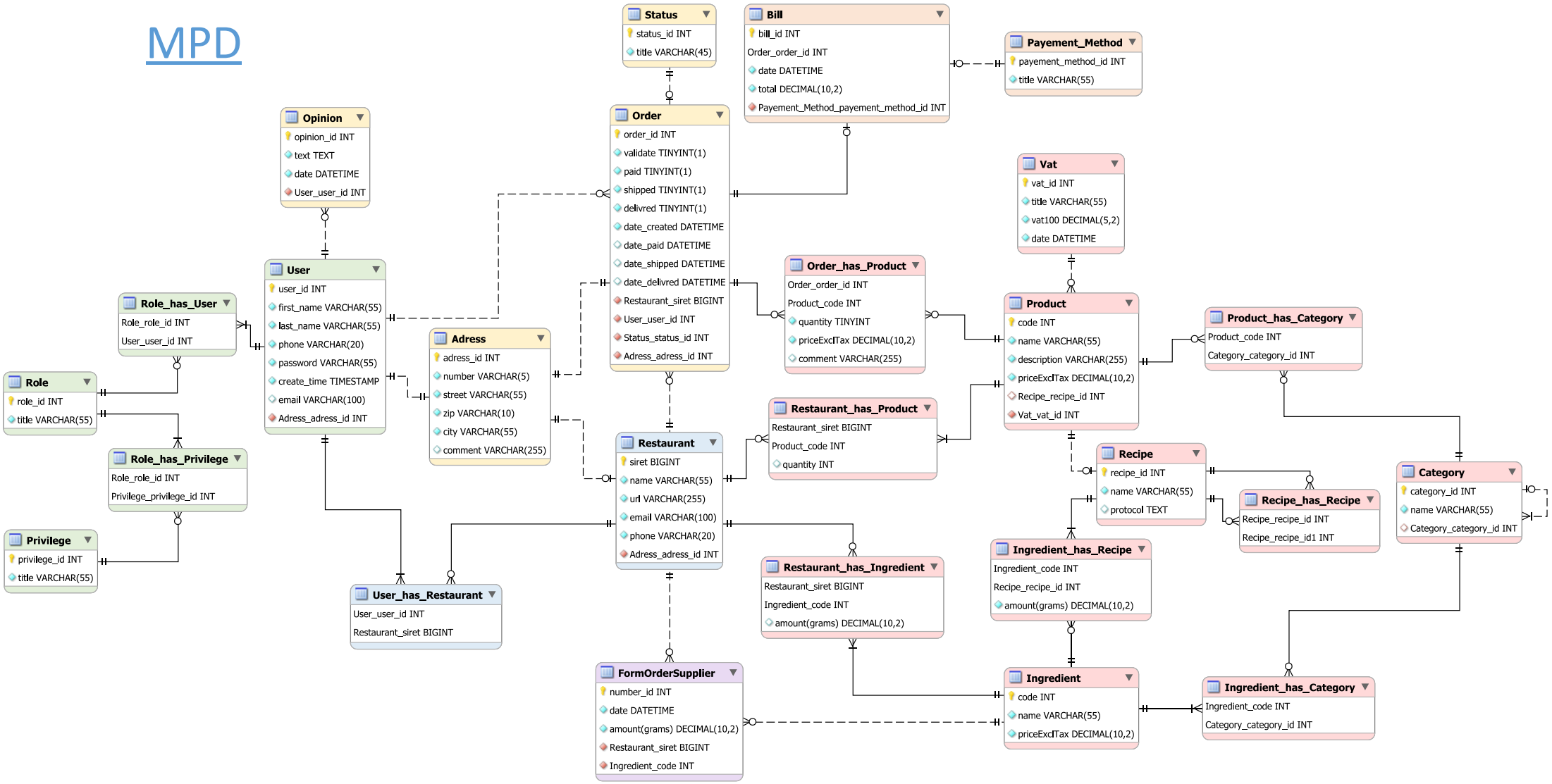
4.2 - Schéma du modèle physique de données

Schéma visible page suivante.
Réalisé avec MySQL Workbench.
Lien vers la documentation : Documentation MySQL Workbench

OC Pizza

IT Consulting & Development

MPD

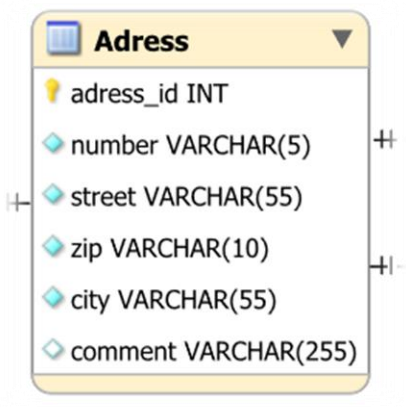


4.3 - Détail des tables

Chaque table est détaillée, ci-dessous, dans deux tableaux :

- Colonne : décrit les types de chaque colonne. La colonne « PK » indique si l'attribut est une clef primaire. La colonne « NN » indique que la valeur NULL n'est pas autorisée. La colonne « AI » précise que la colonne est auto-incrémentée. La mention « FK » précise qu'il s'agit d'une clef étrangère (faisant référence à une autre table).
- Relation : indique la multiplicité (ou cardinalité) de la relation entre la table et celle indiquée dans la colonne « Table », décrit la relation en précisant éventuellement sa nature (jonction).
- Les tables de jonction sont détaillées avec les tables auxquelles elles sont liées.

4.3.1 - Table « Adress »

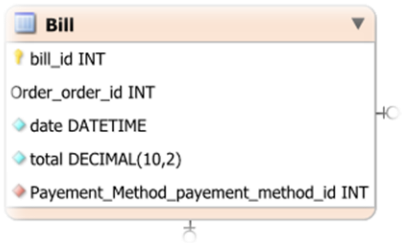


Colonne	Type	PK	NN	AI
adress_id	*INT : référence de la table.*	✓	✓	✓
number	*VARCHAR : 5 charactères autorisés.*		✓	
street	*VARCHAR : 55 charactères autorisés.*		✓	
zip	*VARCHAR : 10 charactères autorisés.*		✓	
city	*VARCHAR : 55 charactères autorisés.*		✓	
comment	*VARCHAR : 255 charactères autorisés.*			

Relation	Table	Description
0..1 – 1	*Order*	*Une adresse est liée à aucune ou plusieurs commandes.* *Une commande est liée à une unique adresse.*
0..1 – 1	*Restaurant*	*Une adresse est liée à aucun ou plusieurs restaurants.* *Un restaurant est lié à une unique adresse postale.*
1 - 1	*User*	*Une adresse de facturation est liée à un unique utilisateur.* *Un utilisateur est lié à une unique adresse de facturation.*

❖ Logique d'utilisation : Avant de créer son panier, le client doit choisir son restaurant. La vérification de l'adresse de livraison se fera lorsque le restaurant devra accepter la commande.

4.3.2 - Table « Bill »

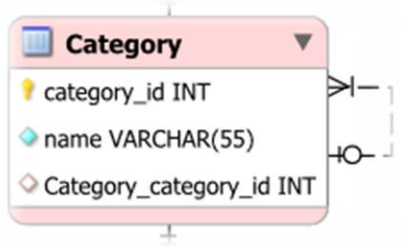


Colonne	Type	PK	NN	AI
bill_id	*INT : référence de la table.*	✓	✓	✓
Order_order_id	*INT : référence de la table.*	✓	✓	
date	*DATETIME : date de création.*		✓	
total	*DECIMAL : montant total.*		✓	
Payement_Method_ Payement_method_id	*FK : renvoie à la table « Payement_Method » et son attribut « Payement_method_id ». Pour identifier le moyen de paiement.*		✓	

Relation	Table	Description
1 – 0..1	*Order*	*Une facture est liée à une unique commande.* *Une commande est liée à aucune ou une facture.*
1 - 1	*Payement_Method*	*Une facture est liée à un unique moyen de paiement.* *Un moyen de paiement est lié à aucune ou une facture.*

- Logique d'utilisation : Chaque facture doit impérativement avoir un numéro unique non réutilisable. La facture est créée au moment du paiement. Eventuellement cette tâche est prise en charge par le service de livraison, une facture est tout de même éditée par le restaurant.

4.3.3 - Table « Category »

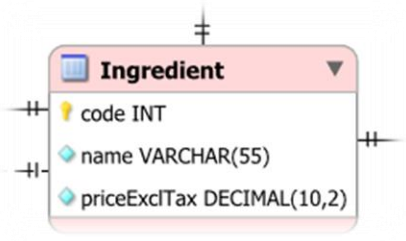


Colonne	Type	PK	NN	AI
category_id	*INT : référence de la table.*	✓	✓	✓
name	*VARCHAR : 55 charactères autorisés.*		✓	
Category_ category_id	*FK : renvoie à la table « Category » et son attribut «* category_id *».*			

Relation	Table	Description
0..1 - 1..*	*Category*	*Une catégorie est liée a aucune ou plusieurs sous-catégories.* *Une sous-catégorie est liée à une unique catégorie.*
Junction	*Ingredient_has_Category*	*Ingredient_has_Category est une table de jonction entre Ingredient et Category* *Une catégorie est liée à aucun ou plusieurs ingrédients.* *Un ingrédient est lié à une ou plusieurs catégories.*
Junction	*Product_has_Category*	*Product_has_Category est une table de jonction entre Product et Category* *Une catégorie est liée à aucun ou plusieurs produits.* *Un produit est lié à aucune ou plusieurs catégories.*

- Logique d'utilisation : Les catégories « produits » et « ingrédients » permettent d'identifier les objets pouvant apparaitre ou non dans le catalogue produit.

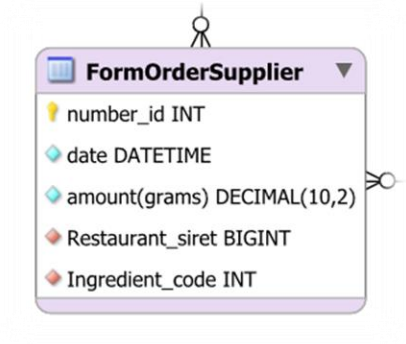
4.3.4 - Table « Ingredient »



Colonne	Type	PK	NN	AI
code	*INT : référence de la table.*	✓	✓	✓
name	*INT : référence de la table.*		✓	
priceExclTax	*DECIMAL : prix Hors taxe.*		✓	

Relation	Table	Description
0..* - 1	*FormOrderSupplier*	*Un ingrédient est lié à aucun ou plusieurs bons de commande.* *Une ligne de bon de commande est lié à un ingrédient.*
Junction	*Ingredient_has_Recipe*	*Ingredient_has_Recipe est une table de jonction entre Ingredient et Recipe.* *Un ingrédient est lié à aucune ou plusieurs recettes.* *Une recette est liée à un ou plusieurs ingrédients.* *La table précise la quantité (en grammes) par ingrédient.*
Junction	*Ingredient_has_Category*	*Ingredient_has_Category est une table de jonction entre Ingredient et Category.* *Un produit est lié à aucune ou plusieurs catégories.* *Une catégorie est liée à aucun ou plusieurs produits.*
Junction	*Restaurant_has_Ingredient*	*Restaurant_has_Ingredient est une table de jonction entre Restaurant et Ingredient.* *Un restaurant est lié à aucun ou plusieurs ingrédients.* *Un ingrédient est lié à un ou plusieurs restaurants.* *La table précise la quantité (en grammes) par ingrédient.*

4.3.5 - Table « FormOrderSupplier »

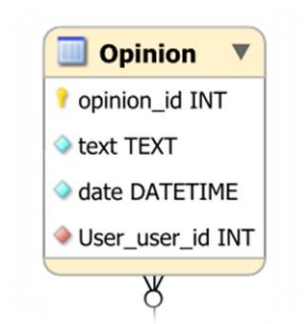


Colonne	Type	PK	NN	AI
number_id	*INT : référence de la table.*	✓	✓	✓
date	*DATETIME : date de création du bon.*		✓	
amount(grams)	*DECIMAL(10,2) : quantité (en grammes) d'ingrédient à commander.*		✓	
Restaurant_siret	*FK : renvoie à la table «* Restaurant *» et son attribut «* siret *». Pour identifier le restaurant à l'origine du bon de commande.*		✓	
Ingredient_code	*FK : renvoie à la table « User » et son attribut « user_id ». Pour identifier l'utilisateur à l'origine de l'avis.*		✓	

Relation	Table	Description
1 – 0..*	*Ingredient*	*Un bon de commande est lié à un unique ingrédient.* *Un ingrédient est lié à aucun ou plusieurs bons de commande.*
1 – 0..*	*Restaurant*	*Un bon de commande est lié à un unique restaurant.* *Un restaurant est lié à aucun ou plusieurs bons de commande.*

- Logique d'utilisation : Le restaurant transmet ce bon au système « fournisseurs » (logiciel).

4.3.6 - Table « Opinion »

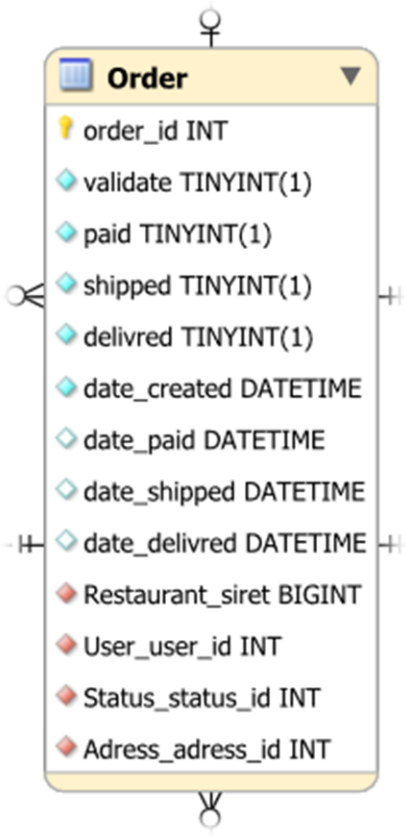


Colonne	Type	PK	NN	AI
opinion_id	*INT : référence de la table.*	✓	✓	✓
text	*TEXT : champ texte de longueur variable.*		✓	
date	*DATETIME : date de création de l'avis.*		✓	
User_user_id	*FK : renvoie à la table « User » et son attribut « user_id ». Pour identifier l'utilisateur à l'origine de l'avis.*		✓	

Relation	Table	Description
1 – 0..*	*User*	*Un avis est lié à un unique utilisateur (client).* *Un utilisateur (client) est lié à aucun ou plusieurs avis.*

- Logique d'utilisation : Dans cette proposition, les avis ne sont pas liés à un restaurant ou une commande spécifique.

4.3.7 - Table « Order »

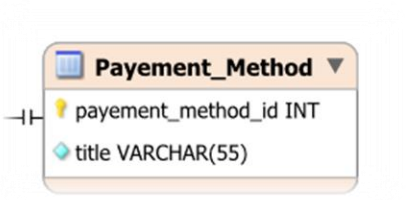


Colonne	Type	PK	NN	AI
order_id	*INT : référence de la table.*	✓	✓	✓
validate	*TINYINT : Type booléen (statut).*		✓	
paid	*TINYINT : Type booléen (statut).*		✓	
shipped	*TINYINT : Type booléen (statut).*		✓	
delivred	*TINYINT : Type booléen (statut).*		✓	
date_created	*DATETIME : date de création.*		✓	
date_paid	*DATETIME : date de paiement.*			
date_shipped	*DATETIME : date de prise en charge par le service de livraison.*			
date_delivred	*DATETIME : date de livraison.*			
Restaurant_ siret	*FK : renvoie à la table « Restaurant » et son attribut « siret ». Pour identifier le restaurant sollicité.*		✓	
User_ user_id	*FK : renvoie à la table « User » et son attribut « user_id ». Pour identifier l'utilisateur à l'origine de la commande.*		✓	
Status_ status_id	*FK : renvoie à la table « Status » et son attribut « status_id ». Pour identifier le statut de la commande.*		✓	
Adress_ adress_id	*FK : renvoie à la table « Adress » et son attribut « adress_id ». Pour identifier l'adresse de livraison.*		✓	

Relation	Table	Description
1 – 0..*	*Adress*	*Une commande est liée à une unique adresse.* *Une adresse est liée à aucune ou plusieurs commandes.*
0..1 – 1	*Bill*	*Une commande est liée à aucune ou une unique facture.* *Une facture est liée à une unique commande.*
Junction	*Order_has_Product*	*Order_has_Product est une table de jonction entre Order et Product.* *Une commande est liée à aucun ou plusieurs produits.* *Un produit est lié à aucune ou plusieurs commandes.* *La table précise la quantité, le prix hors taxes et l'ajout d'un commentaire, par produit.*
1 – 0..*	*Restaurant*	*Une commande est liée à un unique restaurant.* *Un restaurant est lié à aucune ou une commande.*
1 – 1	*Status*	*Une commande est liée à un unique statut.* *Un statut est lié à aucune ou plusieurs commandes.*
1 – 0..*	*User*	*Une commande est liée à un unique utilisateur.* *Un utilisateur est lié à aucune ou plusieurs commandes.*

- Logique d'utilisation : La gestion du panier est dédié au site web.

4.3.8 - Table « Payement_Method »



Colonne	Type	PK	NN	AI
payement_method_id	*INT : référence de la table.*	✓	✓	
title	*VARCHAR : 55 charactères autorisés.*		✓	

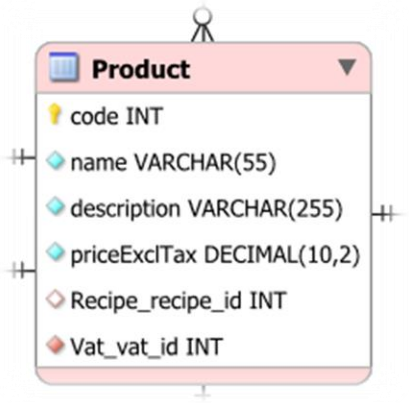
Relation	Table	Description
0..* - 1	*Bill*	*Un moyen de paiement est lié à aucune ou une facture.* *Une facture est liée à un unique moyen de paiement.*

4.3.9 - Table « Privilege »

Colonne	Type	PK	NN	AI
privilege_id	*INT : référence de la table.*	✓	✓	✓
title	*VARCHAR : 55 charactères autorisés.*		✓	

Relation	Table	Description
Junction	*Role_has_Privilege*	*Role_has_Privilege est une table de jonction entre Role et Privilege.* *Un role est lié à aucun ou plusieurs utilisateurs.* *Un utilisateur est lié à au moins un ou plusieurs utilisateurs.*
Junction	*Role_has_User*	*Role_has_User est une table de jonction entre Role et User.* *Un utilisateur est lié à un ou plusieurs rôles.* *Un rôle est lié à aucun ou plusieurs utilisateurs.*

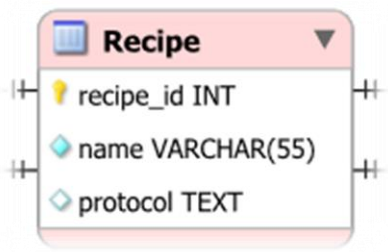
4.3.10 - Table « Product »



Colonne	Type	PK	NN	AI
code	*INT : référence de la table.*	✓	✓	✓
name	*INT : référence de la table.*	✓	✓	
description	*DATETIME : date de création.*		✓	
priceExclTax	*DECIMAL : montant total.*		✓	
Recipe_recipe_id	*FK : renvoie à la table « Recipe » et son attribut « recipe_id ».*			
Vat_vat_id	*FK : renvoie à la table « Vat » et son attribut « vat_id ».*		✓	

Relation	Table	Description
Junction	*Order_has_Product*	*Order_has_Product est une table de jonction entre Order et Product.* *Un produit est lié à aucune ou plusieurs commandes.* *Une commande est liée à aucun ou plusieurs produits.* *La table précise la quantité, le prix hors taxes et l'ajout d'un commentaire, par produit.*
Junction	*Product_has_Category*	*Product_has_Category est une table de jonction entre Product et Category.* *Un produit est lié à aucune ou plusieurs catégories.* *Une catégorie est liée à aucun ou plusieurs produits.*
0..1 – 1	*Recipe*	*Un produit est lié à aucune ou une recette.* *Une recette est liée à un unique produit.*
Junction	*Restaurant_has_Product*	*Restaurant_has_Product est une table de jonction entre Restaurant et Product.* *Un produit est lié à un ou plusieurs restaurants.* *Un restaurant est lié à aucun ou plusieurs produits.* *La table précise la quantité (en grammes) par produit.*
1 – 0..*	*Vat*	*Un produit est lié à une unique T.V.A.* *Une T.V.A. est liée à aucun ou plusieurs produits.*

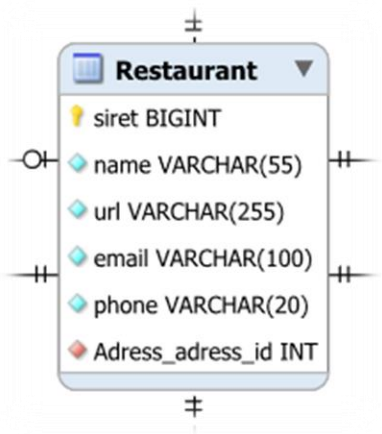
4.3.11 - Table « Recipe »



Colonne	Type	PK	NN	AI
recipe_id	*INT : référence de la table.*	✓	✓	✓
name	*VARCHAR : 55 charactères autorisés.*		✓	
protocol	*TEXT : champ texte de longueur variable.*			

Relation	Table	Description
Junction	*Ingredient_has_Recipe*	*Ingredient_has_Recipe est une table de jonction entre Ingredient et Recipe.* *Un ingrédient est lié à aucune ou plusieurs recettes.* *Une recette est liée à un ou plusieurs ingrédients.* *La table précise la quantité (en grammes) par recette.*
1 – 1	*Product*	*Une recette est liée à un unique produit.* *Un produit est lié à une unique recette.*
Junction	*Recipe_has_Recipe*	*Recipe_has_Recipe est une table de jonction reliant Recipe a elle-même.* *Une recette est liée à aucune ou plusieurs recettes.*

4.3.12 - Table « Restaurant »



Colonne	Type	PK	NN	AI
siret	*BIGINT : le SIRET comporte 11 chiffres.*	✓	✓	
name	*VARCHAR : 55 charactères autorisés.*		✓	
url	*VARCHAR : 255 charactères autorisés.*		✓	
email	*VARCHAR : 100 charactères autorisés.*		✓	
phone	*VARCHAR : 20 charactères autorisés.*		✓	
Adress_ adress_id	*FK : renvoie à la table « Adress » et son attribut « adress_id ». Adresse postale.*		✓	

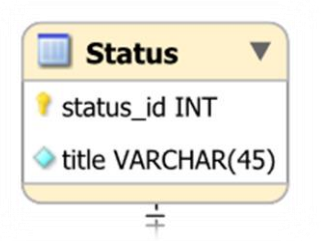
Relation	Table	Description
1 – 1	*Adress*	*Un restaurant est lié à une unique adresse postale.* *Une adresse postale est liée à aucun ou un unique restaurant.*
0..* - 1	*FormOrderSupplier*	*Un restaurant est lié à aucun ou plusieurs bon de commande.* *Un bon de commande est lié à un unique restaurant.*
0..* - 1	*Order*	*Un restaurant est lié à aucune ou plusieurs commandes.* *Une commande est liée à un unique restaurant.*
Jonction	*Restaurant_has_Ingredient*	*Restaurant_has_Ingredient est une table de jonction entre Restaurant et Ingredient.* *Un restaurant est lié à aucun ou plusieurs ingrédients.* *Un ingrédient est lié à un ou plusieurs restaurants.* *La table précise la quantité (en grammes) par ingrédient.*
Jonction	*Restaurant_has_Product*	*Restaurant_has_Product est une table de jonction entre Restaurant et Product.* *Un restaurant est lié à aucun ou plusieurs produits.* *Un produit lié à un ou plusieurs restaurants.* *La table précise la quantité par produit.*
Jonction	*User_has_Restaurant*	*User_has_Restaurant est une table de jonction entre User et Restaurant.* *Un restaurant est lié à aucun ou plusieurs utilisateurs (employé, manager).* *Un employé est lié à un unique restaurant.*

4.3.13 - Table « Role »

Colonne	Type	PK	NN	AI
role_id	*INT : référence de la table.*	✓	✓	✓
title	*VARCHAR : 55 charactères autorisés.*		✓	

Relation	Table	Description
Junction	*Role_has_Privilege*	*Role_has_Privilege est une table de jonction entre Role et Privilege.* *Un role est lié à aucun ou plusieurs utilisateurs.* *Un utilisateur est lié à au moins un ou plusieurs utilisateurs.*
Junction	*Role_has_User*	*Role_has_User est une table de jonction entre Role et User.* *Un utilisateur est lié à un ou plusieurs rôles.* *Un rôle est lié à aucun ou plusieurs utilisateurs.*

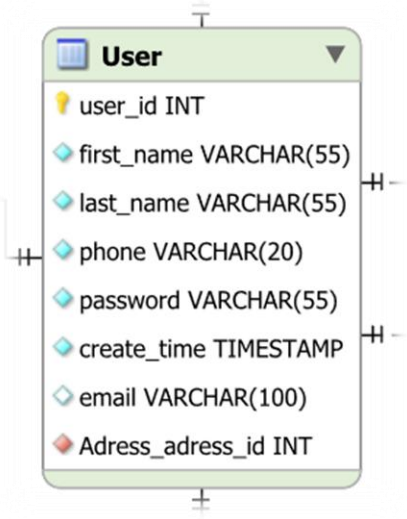
4.3.14 - Table « Status »



Colonne	Type	PK	NN	AI
status_id	*INT : référence de la table.*	✓	✓	✓
title	*VARCHAR : 45 charactères autorisés.*		✓	

Relation	Table	Description
0..* – 1	*Order*	*Un statut est lié à aucune ou plusieurs commandes.* *Une commande est lié à un unique statut.*

4.3.15 - Table « User »



Colonne	Type	PK	NN	AI
user_id	*INT : référence de la table.*	✓	✓	✓
first_name	*VARCHAR : 55 charactères autorisés.*		✓	
last_name	*VARCHAR : 55 charactères autorisés.*		✓	
phone	*VARCHAR : 20 charactères autorisés.*		✓	
password	*VARCHAR : 55 charactères autorisés.*		✓	
create_time	*VARCHAR : 255 charactères autorisés.*		✓	
email	*TIMESTAMP : date de création.*			
Adress_ Adress_id	*FK : renvoie à la table « Adress » et son attribut « adress_id ». Pour la facturation.*		✓	

Relation	Table	Description
	Adress	*Un utilisateur est lié à une unique adresse de facturation.* *Une adresse de facturation est liée à un unique utilisateur.*
	Opinion	*Un utilisateur est lié à aucun ou plusieurs avis.* *Un avis est lié à un unique utilisateur.*
	Order	*Un utilisateur est lié à aucune ou plusieurs commandes.* *Une commande est liée à un unique utilisateur.* *Le protocole de réalisation d'une commande en cas de prise en charge par un employé (ex : commande passée directement au restaurant) est du ressort de la logique métier (à définir avec le client).*
Junction	*Role_has_User*	*Role_has_User est une table de jonction entre Role et User.* *Un utilisateur est lié à un ou plusieurs rôles.* *Un rôle est lié à aucun ou plusieurs utilisateurs.*
Junction	*User_has_Restaurant*	*User_has_Restaurant est une table de jonction entre User et Restaurant.* *Un restaurant est lié à aucun ou plusieurs utilisateurs (employé, manager).* *Un employé est lié à un unique restaurant.*

- Logique d'utilisation : Les clients et employés sont rassemblés dans une unique table « User », la différentiation pouvant se faire par l'attribution de rôle spécifiques. Chaque rôle obtenant des privilèges il sera possible de créer de nouveaux rôles pour répondre à des besoins spécifiques, même ponctuels.

5 - ARCHITECTURE TECHNIQUE

5.1 - Synthèse

Le **système informatique** proposé dans le dossier de dossier de conception fonctionnelle est composé de **parties logicielles** et de **données** qui sont décrites dans la partie suivante et illustrées dans le **diagramme de composants**. L'intérêt du découpage par composants est d'identifier des parties relativement indépendantes. Le diagramme de composants illustre également les **relations** entre ces composants au travers de la visualisation des ports et des dépendances. Il permet d'avoir une vue d'ensemble « statique » sur le projet de développement en cours.

Pour guider la lecture une différentiation est proposée par un code couleur selon les usagers concernés. Cette segmentation n'est bien pas absolue : certaines fonctionnalités peuvent être transverses.

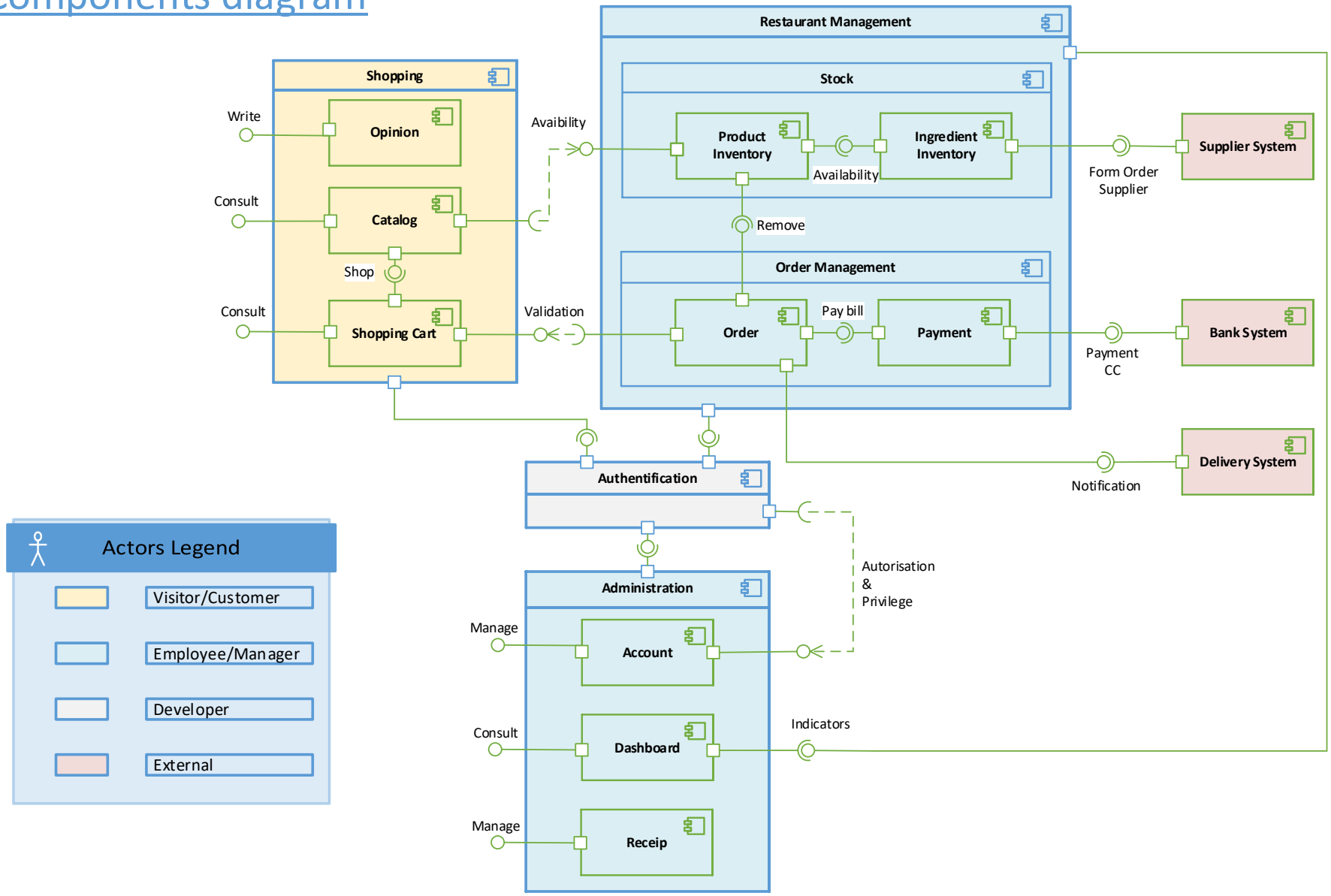
5.2 - Diagramme de composants

Schéma visible page suivante.

OC Pizza

IT Consulting & Development

Components diagram



5.3 - Détail des composants

5.3.1 - Package « Administration »

Le package « **Administration** » est destiné principalement à la direction du groupe et aux responsables de point de vente. Il rassemble le composant « **Accounts** » permettant de gérer les comptes utilisateurs, le composant « **Dashboard** » représentant l'outil affichant les indicateurs et le composant « **Receips** » permettant de gérer l'aide-mémoire.

5.3.2 - Package « Authentification »

Le package « **Authentification** » est le premier à être sollicité lorsqu'un utilisateur souhaite se connecter. Il permet à ce dernier d'obtenir son rôle, donc ses privilèges associés et ainsi de pouvoir accéder à son interface et ses options dédiées.

5.3.3 - Package « Restaurant Management »

Le package « **Restaurant Management** » est destiné aux employés et responsables de point de vente. Il contient l'interface de gestion des stocks, « **Stock** ». Ce package différentie la gestion du stock de produits, « **Product Inventory** » de celle du stock d'ingrédients, « **Ingredient Inventory** », la disponibilité des premiers dépendant des derniers. Ce package contient aussi l'interface de gestion des commandes « **Order Management** ». Il permet aux employés de mettre à jour les stocks, de gérer les commandes (composant « **Order** ») et d'effectuer un encaissement (composant « **Payement** »). Le retrait des produits est fait par le système dès qu'une commande est passée. Ce package communique avec les composants externes pour transmettre un bon de commande d'ingrédients, obtenir une autorisation de paiement et notifier le système de livraison de l'était de la commande. La mise à jour du catalogue produit est faite par le système en fonction de la disponibilité des produits. Une commande devient active lorsque l'information est transmise à partir d'un panier.

5.3.4 - Package « Shopping »

Le package « **Shopping** » correspond à l'interface d'achat et par extension l'interface **client**. Les clients y ont accès à la possibilité de proposer un avis, par le composant « **Opinion** », de consulter le catalogue produit, par le composant « **Catalog** » et en parcourant celui-ci de remplir un panier par le composant « **ShoppingCart** ». Une fois le panier validé, l'information de création de commande est transmise au package « **Restaurant Management** ».

5.3.5 - Composants « Externes »

N'étant pas pris en charge par notre système, on note la présence de trois **composants externes** :

- « **Supplier System** » : logiciel gérant les bons de commande d'ingrédients.
- « **Bank System** » : Api bancaire pour les autorisations de paiement.
- « **Delivery System** » : représentant l'Api de l'entreprise gérant la livraison.

6 - ARCHITECTURE DE DEPLOIEMENT

6.1 - Synthèse

Le **diagramme de déploiement** est la dernière vue « statique **»** des diagrammes **UML**. Il illustre l'utilisation de l'**infrastructure physique** par le système et ses acteurs, ainsi que la nature de leurs **relations**. Il identifie les **éléments matériels** (serveurs) nécessaires et précise, au cœur des nœuds, par le biais de ses composants, quels **logiciels** seront utilisés.

La **pile logicielle** est la suivante :

	Logiciel	Version
Application	Python	3.8
Framework	Django	3.1
Serveur d'application (WSGI)	Gunicorn	20.0
Serveur web	Nginx	1.19
Serveur base de données	MySQL	8.0

Les versions proposées correspondent à celles envisagée au moment de la rédaction du présent document mais sont sujettes à évoluer.

6.2 - Diagramme de déploiement

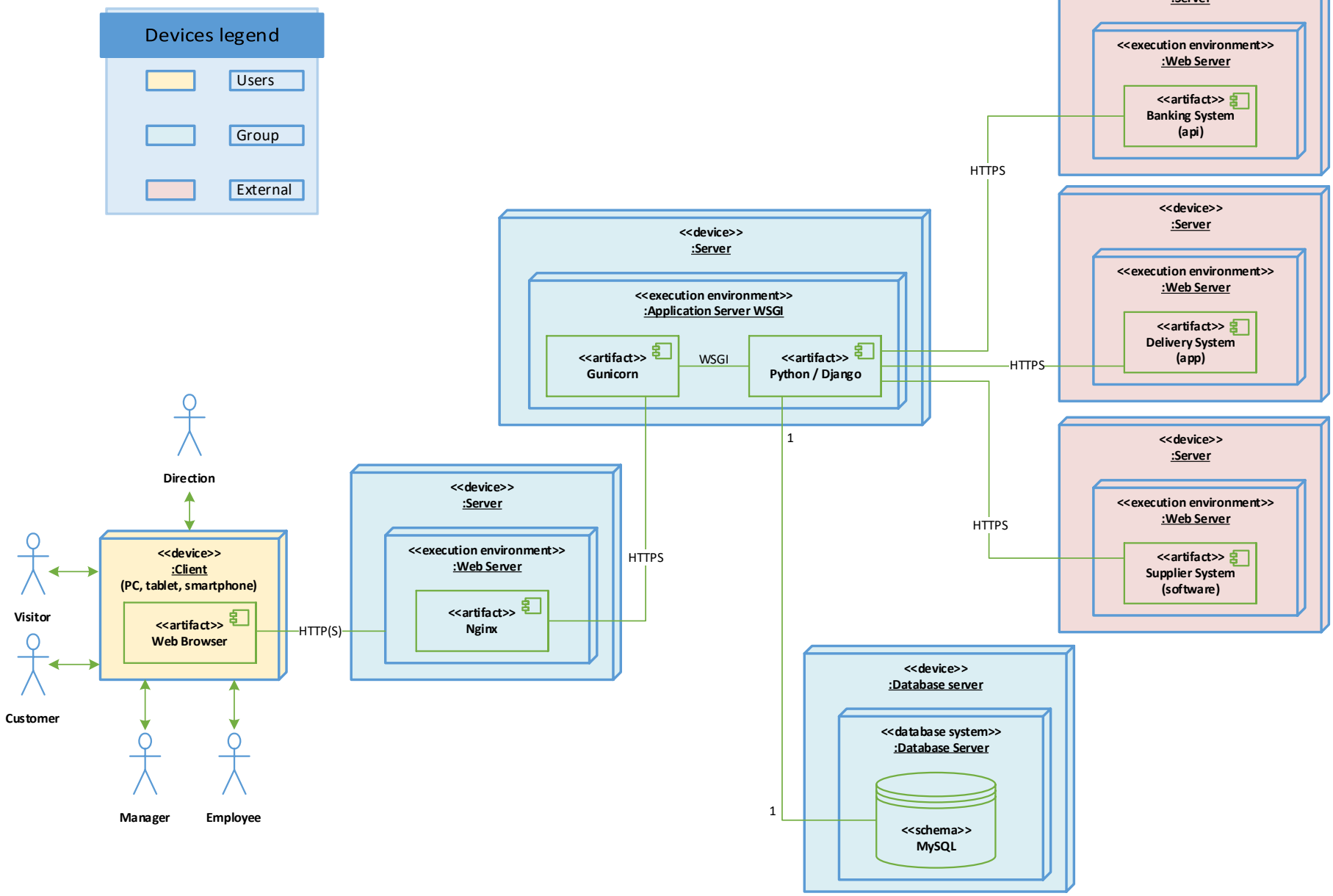
Schéma visible page suivante.

Les terminaux (« devices ») présentés sont regroupés dans trois groupes, seuls ceux identifiés par « group » (couleur bleue) font partit du système que nous développons.

OC Pizza

IT Consulting & Development

Deployement diagram



6.3 - Détail des terminaux

6.3.1 - « External »

Notre système sera amené à communiquer avec des terminaux externes tels que :

- « **Banking System** » : **API** qui sera l'interface avec le système bancaire pour obtenir des **paiements** ou des autorisations de paiement.
- « **Delivery System** » : qui sera l'interface avec le système de livraison, donc selon notre proposition une entreprise externe. Nous utiliserons une application (et/ou API) avec laquelle nous pourrons lui transmettre et recevoir les informations nécessaires à la **prise en charge des commandes** et à la gestion des **paiements à la livraison**. Elle permettra également de suivre en temps réel la **position des livreurs**.
- « **Supplier System** » : ce terminal désigne un logiciel à part entière auquel les restaurants transmettront simplement leurs besoins en ingrédients par le biais de « bon de commandes ». La **gestion des factures « fournisseurs »** et de leurs montant sera effectuée par ce logiciel.

6.3.2 - « Group »

Notre solution proposera donc de mettre en place :

- Un serveur web, **Nginx** : qui sera en charge de recueillir les requêtes **HTTP**(s) transmises par les utilisateurs et de les transmettre à notre serveur d'application.
- Un serveur d'application, **Gunicorn** : qui hébergera les composants logiciels que nous développerons, par le biais de la communication « WSGI » (Web Server Gateway Interface). Il contient la logique métier, cœur de notre programme, rédigé avec le langage **Python** et son framework **Django**.
- Un serveur de base de données, **MySQL** : qui servira à stocker, gérer et extraire les données dans une base de données.

6.3.3 - « Users »

Les usages des utilisateurs pouvant être sur **différents terminaux** nous mettrons en place une solution « **responsive** » : le site d'adaptera et se redimensionnera automatiquement selon les contraintes de résolution des **navigateurs** et de leurs supports : smartphone/tablette/ordinateur. Leurs requêtes seront transmises généralement en HTTP ou HTTPS notamment lors de la connexion.

.

7 - GLOSSAIRE

AI	« Auto Incremental » : attribut auto-incrémenté d'une table SQL.
BDD	Base De Données
Composant	Un composant UML est une unité autonome fournissant un service précis. Sa vocation est d'être réutilisable
Employés	Désigne généralement à la fois, les employés, les responsables de points de vente et la direction du groupe.
FK	« Foreign Key » : clef secondaire d'une table SQL.
Framework	Désigne un ensemble cohérent de composants logiciels structurels, qui sert à créer les fondations ainsi que les grandes lignes de tout ou d'une partie d'un logiciel.
PK	« Primary Key » : clef primaire d'une table SQL.
Pizzaïolo	Préparateur de pizza